Submitted to Globecom 2001

PMM: A Pipelined Maximal-Sized Matching Scheduling Approach for Input-Buffered Switches

Eiji Oki*, *Member, IEEE*, Roberto Rojas-Cessa**, *Student Member, IEEE*, and H. Jonathan Chao**, *Fellow, IEEE*

Abstract

This paper proposes an innovative Pipeline-based Maximal-sized Matching scheduling approach, called PMM, for input-buffered switches. It dramatically relaxes the timing constraint for arbitration with a maximal matching scheme. In the PMM approach, arbitration operates in a pipelined manner. Each subscheduler is allowed to take more than one time slot for its matching. Every time slot, one of them provides the matching result. The subscheduler can adopt a pre-existing efficient maximal matching algorithm such as *i*SLIP and DRRM. PMM maximizes the efficiency of the adopted arbitration scheme by allowing sufficient time for a number of iterations. We show that PMM preserves 100% throughput under uniform traffic and fairness for best-effort traffic as the pre-existing algorithm does.

Keywords

Scheduling, pipeline, input-buffered switch, maximal-sized matching

*Eiji Oki is a corresponding author. He is with NTT Network Service Systems Laboratories, 3-9-11 Midori-cho Musashino-shi, Tokyo 180-8585 Japan. Tel: +81-422-59-3442, Fax: +81-422-59-4549. E-mail: oki.eiji@lab.ntt.co.jp. This work was done while he was a Visiting Scholar at Polytechnic University, Brooklyn, NY.

**Roberto Rojas-Cessa and H. Jonathan Chao are with Polytechnic University, Brooklyn, NY. Email: {rrojas, chao}@kings.poly.edu

I. Introduction

The explosion of Internet traffic has led to a greater need for high-speed switches and routers that have over 1-Tbit/s throughput [14] . Most high-speed packet switching systems adopt a fixed-size cell in the switch fabric. Variable-length packets are segmented into several fixed-sized cells when they arrive, are switched through the switch fabric, and are reassembled into packets before they depart.

There are various types of buffering strategies in switch architectures: input buffering, output buffering, or crosspoint buffering [8], [9], [13]. Input buffering is a cost-effective approach for high-speed switches. This is because input-buffered switches do not require internal speedup or allocate any buffers at each crosspoint. It relaxes memory-bandwidth and memory-size constraints.

In input-buffered switches, it is well known that head-of-line (HOL) blocking limits the maximum throughput to 58.6% in a input-buffered switch with the Fist-In-First-Out (FIFO) structure [4]. A Virtual-Output-Queue (VOQ) structure is used to overcome the HOL-blocking problem [5]. Consider an $N \times N$ input-buffered switch with VOQs at the inputs and a crossbar switch fabric, as shown in Figure 1. A fixed-size cell is sent from any input to any output, provided that no more than one cell is sent from the same input and no more than one cell is received by the same output. Each input i has N VOQs, each of which is denoted as $VOQ(i, j)$, where cells that are destined for output j are stored. The HOL cell in each VOQ can be selected for transmission across the switch in each time slot. Therefore, every time slot, a scheduler has to determine one set of matching.

For a input-buffered switch with VOQs, maximum-sized matching algorithms have been proposed to achieve 100% throughput [3], [6]. Although the maximum-sized matching algorithms provide a maximum match, they suffer from high-computing time complexity. Therefore, it is difficult to implement such algorithms for high-speed switching systems.

Maximal-sized matching algorithms have been proposed as an alternative to maximum matching ones, such as *i*SLIP [5] and Dual Round-Robin Matching (DRRM) [1], [2]. Both algorithms reduce their computing complexity compared with maximum matching ones, and provide 100% throughput under uniform traffic and complete fairness for best-effort

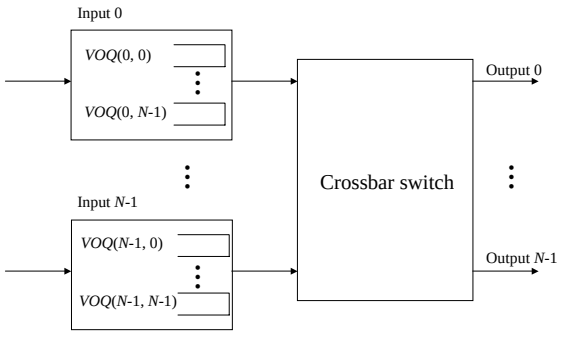

Fig. 1. Input-buffered switch with VOQs.

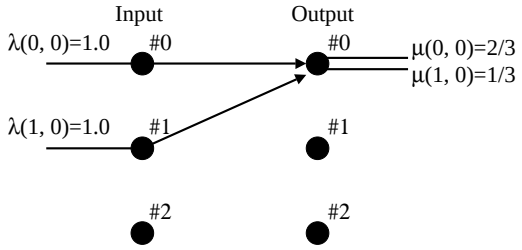

Fig. 2. Example of unfairness for RRGS under unbalanced traffic

traffic. However, they still have a strict constraint that the maximal matching has to be completed within one cell time slot. The constraint is a bottleneck when the switch size increases or a port speed becomes high, because the arbitration time becomes longer than one time slot or the time slot shrinks, respectively. Consider a 64-byte fixed-length cell at a port speed of 40Gbit/s (OC-786). The computation time given for maximal-sized matching is only 12.8ns.

To relax the strict scheduling timing constraint, a pipeline-based scheduling algorithm called Round-Robin Greedy Scheduling (RRGS) was proposed by Smiljanic et al. [10]. Each input has only to perform one round-robin arbitration within one time slot to select one VOQ. N input round-robin operations that select its VOQ as a candidate to be transmitted at time slot T are allocated into the different previous N time slots $\{T - N, T - N + 1, \ldots, T - 1\}$ in a simple cyclic manner so that RRGS can avoid output contention.

However, RRGS is not able to provide max-min fair share for a best-effort service. Let $\lambda(i, j)$ and $\mu(i, j)$ be input offered load to $VOQ(i, j)$ and acceptable transmission rate from $VOQ(i, j)$, respectively. Consider a 3×3 switch, and $\lambda(0, 0) = \lambda(1, 0) = 1.0$ and other input offered loads $\lambda(i, j) = 0$, as shown in Figure 2. According to the RRGS algorithm described in [10], the acceptable transmission rate is obtained as $\mu(0, 0) = 2/3$ and $\mu(1, 0) = 1/3$. This is due to the simple cyclic allocation mechanism. The RRGS operation in this example is described in Appendix A. Thus, when traffic is not balanced, some inputs can unfairly send more cells than others. Smiljanic also proposed weighted-RRGS (WRRGS), which guarantees pre-reserved bandwidth [11]. In WRRGS, however, the fairness problem is not yet solved for best-effort traffic. In addition, every N time-slot cycle, an idle time slot is produced when N is an even number. This means that RRGS does not efficiently utilize the switching capacity.

It is a challenge to find a maximal matching scheduling scheme to meet the following requirements.

- The scheduling time should be relaxed into more than one time slot.

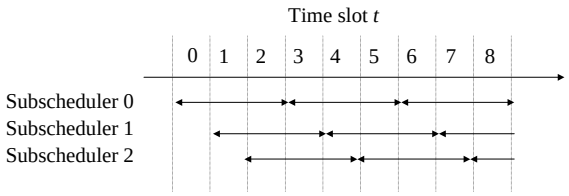

Fig. 3. Timing diagram of PMM with $K = 3$

- High throughput should be provided.
- Fairness should be maintained for best-effort traffic.

This paper presents a solution to these requirements. We introduce an innovative Pipeline-based approach for Maximal-sized Matching scheduling in input-buffered switches, called PMM. Within the scheduler, more than one subscheduler operate in a pipelined manner. Each subscheduler is allowed to take more than one time slot. Every time slot, one of them provides the matching result. The subschedulers can adopt a pre-existing maximal matching algorithm such as *i*SLIP and DRRM, while preserving their properties in the same way as the original non-pipelined version. Therefore, PMM provides 100% throughput under uniform traffic, and maintains fairness for best-effort traffic.

The remainder of this paper is organized as follows. Section II describes the PMM scheme. Section III describes the performance of the PMM scheme. In Section IV summarizes the key points.

II. Pipeline-Based Maximal-Sized Matching (PMM)

The PMM scheme is able to relax the computation time for maximal-sized matching into more than one time slot. A main scheduler consists of N^2 request counters and K subschedulers. Each subscheduler has N^2 request flags. Each subscheduler operates the maximal-sized matching in a pipelined manner, as shown in Figure 3. Each scheduler takes K time slots to complete the matching. In each subscheduler, we assume that one of the pre-existing maximal matching algorithms, DRRM, is adopted to simplify the description below, otherwise stated. PMM described here is also able to adopt other max-min fair share algorithms such as *i*SLIP [5].

We note that DRRM is logically equivalent to *i*SLIP, although each implementation is different [2]. The logical equivalence between DRRM and *i*SLIP can be easily derived in the same ways as a logical equivalence between Parallel Iterative Matching (PIM) and Logical-Equivalent PIM (LE-PIM) was derived by Nong et al. in [7]. Let Θ is an input-traffic matrix. Each element $\theta(i, j)$ expresses input traffic to $VOQ(i, j)$. The performance of DRRM with Θ is equivalent to that of *i*SLIP with Θ^T, where Θ^T is a transposed matrix of Θ.

Several notations used here are defined in the following. A request counter $RC(i, j)$ is associated with $VOQ(i, j)$. The value of $RC(i, j)$ is denoted as $C(i, j)$, where $0 \leq C(i, j) \leq$

L_{max}. L_{max} is the maximum VOQ occupancy. $C(i,j)$ expresses the number of accumulated requests associated with $VOQ(i,j)$ that have not been sent to any subschedulers. Each request flag $RF(i,j,k)$ is associated with $VOQ(i,j)$ and subscheduler k, where $0 \leq k \leq K-1$. The value of $RF(i,j,k)$ is denoted as $F(i,j,k)$, where $0 \leq F(i,j,k) \leq 1$. $F(i,j,k) = 1$ means that input i has a request to output j in subscheduler k. $F(i,j,k) = 0$ means that input i has no request to output j in subscheduler k. At initial time, $C(i,j)$ and $F(i,j,k)$ are set to zero.

PMM operates as follows.

- Phase 1: When a new cell enters $VOQ(i,j)$, the counter value of $RC(i,j)$ is increased: $C(i,j) = C(i,j) + 1$.
- Phase 2: At the beginning of every time slot t, if $C(i,j) > 0$ and $F(i,j,k) = 0$, where $k = t \mod K$, $C(i,j) = C(i,j) - 1$ and $F(i,j,k) = 1$. Otherwise, $C(i,j)$ and $F(i,j,k)$ are not changed.
- Phase 3: At $Kl + k \leq t < K(l+1) + k$, where l is an integer, subscheduler k operates the maximal-sized matching according to the adopted algorithm.
- Phase 4: At the end of every time slot t, subscheduler k, where $k = (t - (K-1)) \mod K$, completes the matching. When input-output pair (i,j) is matched, $F(i,j,k) = F(i,j,k) - 1$. In this case, the HOL cell in $VOQ(i,j)$ is sent to output j at the next time slot.[1] When input-output pair (i,j) is not matched, $F(i,j,k)$ is not changed.

Whenever a condition associated with any phase is satisfied, the phase is executed by the main scheduler.

To apply the DRRM algorithm as a matching algorithm in subscheduler k in PMM, we use $F(i,j,k)$ instead of VOQ requests as described in the DRRM scheme [1], [2]. Each subscheduler has its own round-robin pointers. The pointers in subscheduler k are influenced by the results only from subscheduler k, not from other subschedulers. The operation of DRRM in subscheduler k is the same as that of the non-pipelined DRRM scheme.

III. Performance

This section describes throughput and delay performance of PMM under uniform traffic, and the fairness for best-effort traffic. In addition, the effect of the PMM scheduling relaxation is described.

A. Throughput

PMM that adopts the DRRM algorithm in the subschedulers provides 100% throughput under uniform traffic.

The reason is as follows. Consider the input load as 1.0. If some inputs cannot send cells, outstanding requests are maintained in each subscheduler. In other words, $F(i,j,k) = 1$. As a result, $C(i,j)$ is not always decremented in phase 2 and increased in phase 1. Since $C(i,j)$ reaches a large enough value to be always satisfied with $C(i,j) > 0$, $F(i,j,k) = 1$

[1]This ensures that cells from the same VOQ are transmitted in sequence, even if $L(i,j) - C(i,j) > 1$, where $L(i,j)$ is the occupancy of $VOQ(i,j)$. Note that $L(i,j) - C(i,j) = \sum_{k=0}^{K-1} F(i,j,k)$.

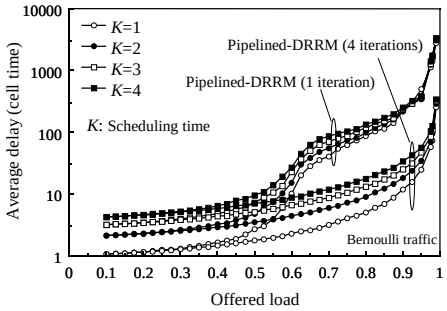

Fig. 4. Delay dependency on scheduling time K for Bernoulli traffic (switch size $N = 32$)

is kept in phase 2.[2] In this situation with $F(i, j, k) = 1$ at any t in phase 3, subscheduler k that adopts the DRRM algorithm provides a complete matching result every K time slot due to the desynchronization effect of input and output arbiters, as described in [2].[3]

Thus, PMM preserves the throughput advantage of DRRM.

B. Delay

The scheduling time K does not significantly degrade delay performance, as shown in Figure 4. Bernoulli arrival process is used for the input traffic. In this evaluation, we include absolute delay caused by the scheduling time in the delay performance. When K increases, requests from $RC(i, j)$ are distributed to $RF(i, j, k)$ associated with each subscheduler k. Therefore, the desynchronization effect becomes less efficient with K at the light traffic load. At the heavy traffic load, the delay dependency on K becomes negligible. Therefore, K does not affect delay performance for a practical use.

The delay performance is improved with more iterations. Since PMM relaxes the scheduling timing constraint, a large number of iterations is not a bottleneck even when the switch size increases or a port speed becomes fast, compared with the non-pipelined algorithm, as will be described in subsection III-D. Note that we showed the delay performance with one and four iterations because there is no measurable improvement with more than 4 iterations.

The above observations for $N = 32$ are applied to different switch sizes. Figure 5 shows the same delay tendency as that in Figure 4.

Figure 6 shows that K does not affect delay performance for a practical use even when a bursty arrival process is considered. We assume that the burst length is exponentially distributed as the burst traffic. In this evaluation, the average burst length is set to be 10.

[2]Although $F(i, j, k)$ becomes 0 in phase 4 when a request is granted, $F(i, j, k)$ is always changed to 1 in phase 2.

[3]We note that the pointer desynchronization is achieved within the same subscheduler. There is no relationship of pointers among different subschedulers.

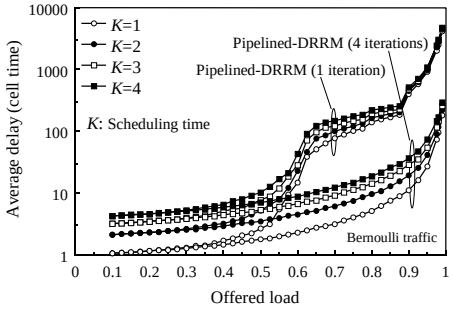

Fig. 5. Delay dependency on scheduling time K for Bernoulli traffic (switch size $N = 64$)

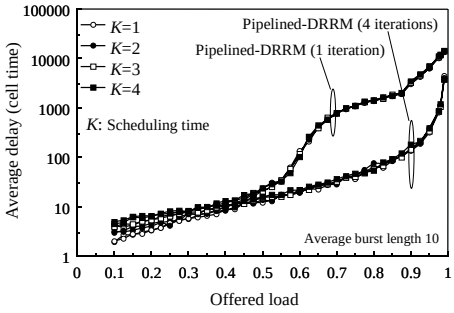

Fig. 6. Delay dependency on scheduling time K for bursty traffic (switch size $N = 32$)

C. Fairness

Since we adopt a round-robin-based algorithm in subscheduler k, subscheduler k can maintain a fair scheduling. Therefore, PMM provides max-min fair share for best-effort traffic.

D. Scheduling Timing Relaxation

Figure 7 shows the effect of the PMM scheduling timing relaxation. We assume that a cell size, L_{cell}, is 64x8 bits. Let the allowable arbitration time per iteration, a port speed, and the number of iterations, be T_{arb}, C, and I. T_{arb} is given by,

$$T_{arb} = \frac{KL_{cell}}{CI}. \tag{1}$$

T_{arb} decreases with I and C, but increases with K. In the non-pipelined DRRM scheme, K is 1 as a special case of PMM in Eq. (1). In the non-pipelined DRRM, when C=40Gbit/s and $I = 4$, T_{arb}=3.2ns. Under this timing constraint, it is difficult to implement round-robin arbiter that supports large N in hardware by using available CMOS technologies, in

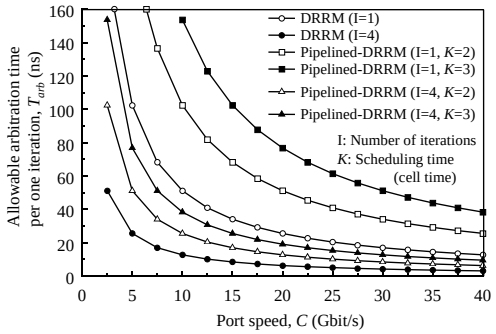

Fig. 7. Effect of PMM scheduling timing relaxation

which, for example, typical gate-delay time is about 100ps [12]. On the other hand, PMM can expand T_{arb} by increasing K. When C=40Gbit/s, $I = 4$ and $K = 3$, T_{arb}=9.6ns. Therefore, PMM achieves the desired number of iterations even when N increases or C becomes fast.

IV. Conclusions

This paper has proposed a Pipeline-based Maximal-sized Matching scheduling approach, called PMM, for input-buffered switches. It dramatically relaxes the scheduling timing constraint. Each subscheduler is allowed to take more than one time slot. Every time slot, one of them provides the matching results. The subscheduler can adopt a pre-existing efficient maximal matching algorithm such as *i*SLIP or DRRM. PMM maximizes the efficiency of the adopted arbitration scheme by allowing sufficient time for a number of iterations. We showed that PMM preserves 100% throughput under uniform traffic and keeps fairness for best-effort service as the pre-existing algorithm does, while ensuring that cells from the same VOQ are transmitted in sequence. Thus, PMM is a solution to a maximal-sized matching scheduler for input-buffered switches when the switch size increases or a port speed becomes high such as OC-768.

References

[1] H. J. Chao and J-S Park, "Centralized Contention Resolution Schemes for a Large-Capacity Optical ATM switch," Proc. IEEE ATM Workshop'97, Fairfax, VA, May 1998.

[2] H. J. Chao, "Saturn: A Terabit Packet Switch Using Dual Round-Robin," IEEE Commun. Mag., vol. 38, no. 12 pp. 78-84, Dec. 2000.

[3] J. E. Hopcroft and R. M. Karp, "An Algorithm for Maximum Matching in Bipartite Gragphs," Soc. Ind. Appl. Math. J. Comptation, vol. 2, pp. 225-231, 1973.

[4] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," IEEE Trans. Commun., vol. COM-35, pp. 1347-1356, 1987.

[5] N. Mckeown, "The *i*SLIP Scheduling Algorithm for Input-Queued Switches," IEEE/ACM Trans. Networking, vol. 7, no. 2, pp. 188-200, Apr. 1999.

[6] N. Mckeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switches," IEEE Trans. Commum., vol. 47, no. 8, pp. 1260-1267, Aug. 1999.

T_0	T_1	T_2	T_3	T_4	T_5	T_6
		I_1 for T_5	I_2 for T_5	I_0 for T_5	I_1 for T_8	I_2 for T_8
	I_2 for T_4	I_0 for T_4	I_1 for T_4	I_2 for T_7	I_0 for T_7	I_1 for T_7
I_0 for T_3	I_1 for T_3	I_2 for T_3	I_0 for T_6	I_1 for T_6	I_2 for T_6	I_0 for T_9

Fig. 8. Timing diagram of RRGS switch

[7] G. Nong, J. K. Muppala, and M. Hamdi, "Analysis of Nonblocking ATM Switches with Multiple Input Queues," *IEEE/ACM Trans. Networking*, vol. 7, no. 1, Feb. 1999, pp.60-74.
[8] G. Nong and M. Hamdi, "On the Provision of Quality-of-Service Guarantees for Input Queued Switches," *IEEE Commun. Mag.*, vol. 38, no. 12, Dec. 2000, pp.62-69.
[9] E. Oki, N. Yamanaka, Y. Ohtomo, K. Okazaki, and R. Kawano, "A 10-Gb/s (1.25 Gb/s × 8) 4 × 2 0.25-μm CMOS/SIMOX ATM Switch Based on Scalable Distributed Arbitration," IEEE J. Solid-State Circuits, vol. 34, no. 12. pp.1921-1934, Dec. 1999.
[10] A. Smiljanic, R. Fan, and G. Ramamurthy, "RRGS-Round-Robin Greedy Scheduling for Electronic/Optical Terabit Switches," Proc. IEEE Globecom'99, pp. 1244-1250.
[11] A. Smiljanic, "Flexible Bandwidth Allocation in Terabit Packet Switches," Proc. IEEE Workshop on High Performance Switching and Routing 2000, pp. 233-239, 2000.
[12] Texas Instruments, "GS40 0:15-μm CMOS, Standard Cell/Gate Array," http://www.ti.com/, version 0.2, May 2000.
[13] J. Turner and N. Yamanaka, "Architectural Choices in Large Scale ATM Switches," IEICE Trans. Commun. vol. E81-B, no. 2, pp. 120-137, Feb. 1998.
[14] N. Yamanaka, E. Oki, S. Yasukawa, R. Kawano, and K. Okazaki, "OPTIMA: Scalable, Multi-Stage, 640-Gbit/s ATM Switching System Based on Advanced Electronic and Optical WDM Technologies," IEICE Trans. Commun., vol. E83-B, no. 7, pp. 1488-1496, 2000.

Appendix

I. Unfairness of RRGS

Before we explain why RRGS suffer from the unfairness problem, we describe briefly the RRGS algorithm. Figure 8 shows an example of the RRGS algorithm. Let us consider 3×3 switch. We denote an round-robin arbiter associated with input i as I_i. At time slot T_0, I_0 chooses a candidate to be transmitted at time slot T_3. At time slot T_1, I_1 and I_2 choose their own candidates to be transmitted at time slot T_3 and T_4, respectively. At time slot T_2, I_2, I_0, and I_1 choose their own candidates to be transmitted at time slot T_3, T_4, and T_5, respectively. In this way, at each time slot, each input arbiter chooses one candidate to be transmitted at different time slot from other arbiters. If we focus on a certain transmission time slot, for example, T_3, the candidates are selected at T_0, T_1, and T_2.

Next, we explain the unfairness problem of RRGS. We consider input traffic as shown in Figure 2. The results obtained by RRGS are shown in Figure 9 At time slot T_0, I_0 chooses

T_0	T_1	T_2	T_3	T_4	T_5	T_6
		VOQ(1,0) for T_5	*Not selected*	*Not selected*	*VOQ(1,0)* for T_8	*Not selected*
	Not selected	*VOQ(0,0)* for T_4	*Not selected*	*Not selected*	*VOQ(0,0)* for T_7	*Not selected*
VOQ(0,0) for T_3	*Not selected*	*Not selected*	*VOQ(0,0)* for T_6	*Not selected*	*Not selected*	*VOQ(0,0)* for T_9

Fig. 9. Candidates selected by RRGS

$VOQ(0,0)$ as a candidate to be transmitted at time slot T_3. Output 0 for T_3 is reserved. At time slot T_1, I_1 and I_2 try to choose their own candidates to be transmitted at time slot T_3 and T_4, respectively. However, I_1 cannot select any candidate for T_3 because there is only one non-empty $VOQ(1,0)$ and output 0 for T_3 is reserved. I_2 cannot select any candidate for T_4 because there is no non-empty VOQ at input 2. As a result, output 0 for T_4 is not reserved at T_1. At time slot T_2, I_0 and I_1 choose $VOQ(0,0)$ and $VOQ(1,0)$ as candidates to be transmitted at time slot T_4 and T_5, respectively. As you can see Figure 9, every three-time-slot cycle, $VOQ(0,0)$ is selected twice while $VOQ(1,0)$ is once. Thus, when traffic is not balanced in the example shown in Figure 2, input 0 can unfairly send twice more cells than input 1.